



NAVAL POSTGRADUATE SCHOOL Monterey, California



THESIS



Multicriteria Integer Zero-One Programming:
A Tree-Search Type Algorithm

by

Aggelos Konstantinou Simopoulos

December 1977

Thesis Advisor:

S. T. Holl

Approved for public release; distribution unlimited.

REPORT DOCUMENT	READ INSTRUCTIONS BEFORE COMPLETING FORE				
. REPORT NUMBER	2. GOVT ACCESSION	N NO. 3. RECIPIENT'S CATALOG NUMBER			
		TYPE OF REPORT & PERIOD COV			
Multicriteria Integer Z	oro-Ono Brogram				
ming: A Tree-Search Ty		Master's Thesis December 1977			
ming. A liee-beaten ly	pe Algorician,	6. PERFORMING ORG. REPORT NUM			
. AUTHOR(e)		S. CONTRACT OR GRANT NUMBER			
Aggelos Konstantinou/Si	mopoulos				
. PERFORMING ORGANIZATION NAME AND	ADDRESS/	10. PROGRAM ELEMENT, PROJECT, AREA & WORK UNIT NUMBERS			
Naval Postgraduate Scho	ool /				
Monterey, California 93					
1. CONTROLLING OFFICE NAME AND ADDR	ESS (Dec 12. REPORT DATE			
Naval Postgraduate Scho		13. NUMBER OF PAGES			
Monterey, California 93		12) 146 P.			
14. MONITORING AGENCY NAME & ADDRESS	il different from Controlling Off	160) 18. SECURITY CLASS. (of this report)			
Naval Postgraduate Scho	001	Unclassified			
Monterey, California 93		ISA. DECLASSIFICATION/DOWNGRA			
		SCHEDULE			
Approved for public rel		ion unlimited.			
	ease; distribut:				
Approved for public rel	ease; distribut:				
Approved for public rel	ease; distribut:				
Approved for public rel	ease; distribut:				
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetroe) 18. SUPPLEMENTARY NOTES	ease; distribut: st entered in Block 20, if difference concern and identify by block m	ant from Report)			
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetract 8. SUPPLEMENTARY NOTES 9. KEY WORDS (Continue on revoce elde if necond) Optimization Com	ease; distribut:	ant from Report)			
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetra- 18. SUPPLEMENTARY NOTES 9. KEY WORDS (Continue on revorce elde if not Optimization Com Multiobject Efficiency	ease; distribut: st entered in Block 20, if difference concern and identify by block m	ant from Report)			
Approved for public rel 17. DISTRIBUTION STATEMENT (of the abotron 18. SUPPLEMENTARY NOTES 19. KEY WORDS (Continue on reverse elde II not Optimization Com Multiobject	ease; distribut: st entered in Block 20, if difference concern and identify by block m	ant from Report)			
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetra- 18. SUPPLEMENTARY NOTES 9. KEY WORDS (Continue on revorce elde if not Optimization Com Multiobject Efficiency	ease; distribut: st entered in Block 20, if different coccess and identify by block ma	ant from Report)			
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetract 8. KEY WORDS (Continue on revoce elde if no Optimization Commultiobject Efficiency Algorithm 10. ASSTRACT (Continue on revoce elde if no on the continue on revoce elde if no other continue on revoce elde if no ot	coccery and identify by block ma	ant from Report)			
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetract 8. KEY WORDS (Continue on revoce elde if no Optimization Commultiobject Efficiency Algorithm 10. ASSTRACT (Continue on revoce elde if no on the continue on revoce elde if no other continue on revoce elde if no ot	coccer and identify by block manager	matery programming problems			
Approved for public rel 7. DISTRIBUTION STATEMENT (of the abetra- 18. SUPPLEMENTARY NOTES 9. KEY WORDS (Continue on reverse side if no- Optimization Com Multiobject Efficiency Algorithm 10. Asstract (Continue on reverse side if no- An algorithm for ze	coccer and identify by block and puter Program ero-one integer processed in the second second in the second second in the second secon	mt from Report) mbor) programming problems is developed, implemen			

DD | FORM | 1473 | EDITION OF | NOV 45 IS OBSOLETE | S/N 0102-014-6401 | 251 450

mality must be replaced with that of efficiency. A solution is said to be efficient if (1) it satisfies the constraints and (2) no other solution satisfying them scores as well with respect to all objective functions and better with respect to at least one of them. In the presented algorithm, the problem variables are partitioned into two second those whose coefficients in the objective functions are all of the same sign, and the remainder. A tree search implicit enumeration algorithm based on this partition is developed and computational results are presented.

Approved for public release; distribution unlimited.

MULTICRITERIA INTEGER ZERO-ONE PROGRAMMING: A TREE-SEARCH TYPE ALGORITHM

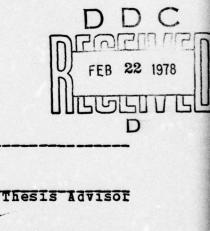
by

Aggelos Konstantinou Simopoulos
Major Hellenic Army
B.S., Military Academy of Greece, 1959
M.S., Technical University of Athenes, 1965

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the
NAVAL POSTGRADUATE SCHOOL
December 1977



Approved by:

Author:

John 7. Thee

Second Feader

RTIS	White Section
200	Buff Section
WHANNOUNC	ED C
JUSTIFICAT	ion
84	
DISTRIBUT	TION/AVAILABILITY CODES
DISTRIBUT	AVAIL, and/or SPECIAL

Chairman, Department of Computer Science
Chairman, Department of Operations research
Dean of Information and Folicy Sciences

and the second of the second o

ABSTRACT

An algorithm for zero-one integer programming problems with more than one objective functions is implemented developed, and tested. Por multiobjective problem the notion of optimality must be replaced with that of efficiency. A solution is said to be efficient if (1) it satisfies the constraints and (2) no other solution satisfying them scores as well with respect to all objective functions and better with respect to at least one of them. the presented algorithm, the problem variables are partitioned into two sets; those whose coefficients in the objective functions are all of the same sign, the remainder. A tree search implicit enumeration algorithm based on this partition is developed and computational results are presented.

TABLE OF CONTENTS

ī.	INI	ROD	UCTI	ON.						•		•		•		. 7
II.	THE	MU.	LTIO	BJEC	TIV	E FU	NCI	ION	P	ROE	BLE	M				.11
	A.	DIF	FERE	NCES	WI	гн т	HE	ONE	-01	BJE	CI	!IV	Æ			
		FUN	CTIO	N PR	OBL	EM .				1	•		•	٠	•	.12
	в.	DEA	LING	WIT	H NI	EGAT	IVE	СО	EFI	FIC	:IE	NI	's	IN		
		SOM	E OF	THE	ОВ	JECI	IVE	FU	NC:	ric	NS					.13
	c.	FOR	MULA	TION	OF	THE	PR	OBL	EM	•						.16
III.	SMA	LL	SCAL	E PR	OBL	EMS.	A	N E	XAI	MPI	E					.19
	A.	TOT.	AL E	NUME	RAT	ION.				٠						.19
	в.	THE	TRE	E-SE	ARCI	H PR	OCE	DUR	E.							.22
IV.	LAF	GE :	SCAL	E PR	OBLI	EMS.									•	. 27
	Α.	AN	ADD	ITIV	E AI	LGOR	ITH	м.								. 27
	в.	IM	PLEM	ENTA	TIO	N US	ING	тн	E	COM	IPU	TE	R			.34
Append																
LIST																
INITIA																
INITIA	AL L	LOT	KIBU'	LION	LI	or .										. 40

ACKNOWLEDGEMENT

I would like to express my appreciation to my thesis advisor LCDR Stephen T. Holl for his continued support and valuable comments and suggestions that were instrumental in improving the readability and contents of this thesis.

Also I would like to thank my thesis second reader Professor Gordon Brandley for his direction and guidance.

I. INTRODUCTION

Before addressing the multiobjective function problem, the tree-search method for one objective function problems will be reviewed. Much work has been done in this area, and different algorithms are described in references [1,2,3,4,5,6,7,8].

The problem generally is formulated as follows:

min
$$Z = c_1 x_1 + c_2 x_2 + \dots + c_n x_n$$

s.t. $a_1 x_1 + a_2 x_2 + \dots + a_n x_n \ge b_n i = 1, 2, \dots, m$
and $x_1 + a_2 x_2 + \dots + a_n x_n \ge b_n i = 1, 2, \dots, m$
and $x_1 + a_2 x_2 + \dots + a_n x_n \ge b_n i = 1, 2, \dots, m$

By reassigning subscripts and applying suitable transformations on the variables the problem can always be transformed to meet the following additional requirements.

- a) $c \ge 0$ for all j. (If some c < 0 then we substitute x = 1-y).
- b) c >c if 1>k.

For a zero-one integer program there are 2^{n} candidate sclutions. All these solutions are ordered in a diagram as shown in Fig. 1 for n=4.

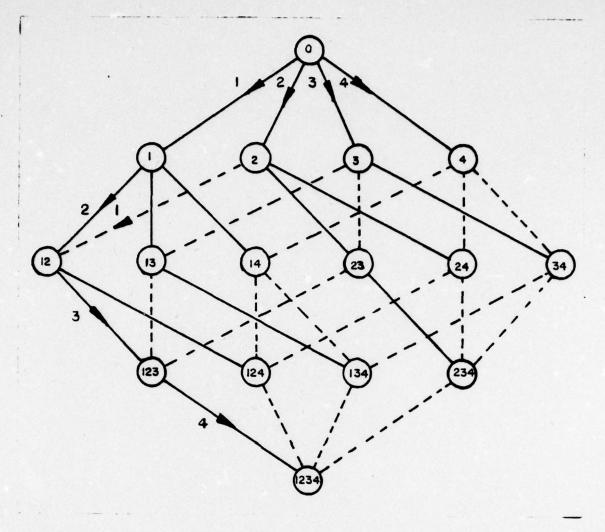


Figure 1.

Each node in the graph of Fig. 1 represents a candidate solution. Inside each node there are indices indicating the solution with x = 1 for these indices and x = 0 otherwise. An j index i is also associated with every arc. This index indicates the variable x = 1 for the node where the arc terminates, and x = 0 for the node where the arc starts. Values of all other variables are the same in both nodes.

Define level i of the graph to be the set of nodes which have i digits representing them. By convention level 0 is the level which has only the node 0. It is easily verified that if there are n variables the highest level will be level n.

Note that level i contains $\binom{n}{i}$ nodes and that there is a symmetry in the structure of the graph so that the level n/2, for n even, or the level $(n/2\pm1/2)$, for n odd, have the higgest number of nodes and this number decreases symmetrically as we go from the middle to the lowest and highest levels.

If there exists a chain from a node Ni to a node Nj, then Ni is said to be predecessor of node Nj and Nj is said to be successor of node Ni. All solutions are partially ordered by the predecessor-successor relationship.

One solution is said to "dominate" another if the objective value associated with the first is better than that associated with the second.

Since x takes the value 0 or 1, the value of the objective function Z is the sum of these coefficients c for which x is 1; also since c ≥0 for all j, the nodes in j higher numbered levels represent worse (greater) values for Z than their predecessors do. Consequently if a solution is feasible or if it is dominated by another feasible solution, there is no need to test its successors since they are dominated.

For example consider the node 1 in level 1. Its Z value

is c_1 , and if this solution is feasible it dominates its successors in level 2, namely nodes 12 with $Z=c_1+c_2$, 13 with $Z=c_1+c_3$ and node 14 with $Z=c_1+c_4$, and their successors in level 3 (nodes 123, 124 and 134) and in level 4 (node 1234).

Another bounding relation appears from the fact that the objective function is formulated in an increasing order of the values of the coefficients c. So for example if c < c j solution 2 dominates solution 3 and solution 24 dominates solution 34 even though these solutions are not related to each other with a predecessor-successor relationship.

In references [7,8] the interested reader will find example problems and more details for the one-objective problem, tree-search type algorithms.

· The way

II. THE MULTIOBJECTIVE FUNCTION PROBLEM

When there are more than one objective function, the notion of optima must be replaced with that of efficiency.

A solution is said to be "efficient" if:

- (1). It satisfies the constraints, and
- (2). No other solution which also satisfies all cf the constraints scores at least as well with respect to all criteria and better with respect to at least one of them.

A single objective implicit enumeration problem will have a unique optimum criteria value, but a multicriterion problem can be expected to have more than one set of efficient criteria. For example consider a problem with two "minimizing" objective functions, Z and W. There may exist two solutions such that Z¹<Z² and W¹>W²; in other words solution (1) is better for the objective function Z and worse for the objective function W. This means that both solutions must be considered in the choice of the final solution. Reference [9] gives an approach to this type of problem.

This thesis addresses the problem of finding all cf the efficient solutions, using a tree-search type algorithm.

A. DIFFERENCES WITH THE ONE-OBJECTIVE FUNCTION PROBLEM

Formulation of the problem for the multiobjective case is as follows:

min
$$z = c$$
 $x + c$ $x + \cdots + c$ x $i = 1, 2, \dots, p$

s.t. $a_{i1} x + a_{i2} x + \cdots + a_{in n} x \ge b$ $i = 1, 2, \dots, m$

and x takes the values 0 or 1 for all j.

Clearly the constraints have the same formulation as before; and the only difference from the single objective case is that now there is more than one objective function. Because of this difference the problem can not be formulated in increasing order of magnitute of the coefficients c ij to illustrate that consider the following two objective functions:

$$Z=3x_1+4x_2+5x_3+...$$

 $W=4x_1-3x_2+5x_3+...$

It is clear that reordering W in an increasing order of the coefficients c destroys the ordering in the objective ij

The second tactic the one-objective function algorithms use to reduce testing, is the formulation of the problem so that $c \ge 0$ for all j. Unfortunately no transformation can

make c_{ij} 20 for all j and for all i; in the example above if one substitutes $x_2=1-y_2$ in order to make $c_{22}>0$, then there is an opposite effect in the first objective function, making $c_{12}<0$. Only if $c_{ij}\leq0$ for all i is this transformation possible.

So in the general case one cannot have positive coefficients in all objective functions, and algorithms for the multiobjective problem must address this greater generality.

B. DEALING WITH NEGATIVE COEFFICIENTS IN SOME OF THE OBJECTIVE FUNCTIONS

Return now to the graph of Fig. 1, which has been constructed for the one-objective case, and study the relationship between the nodes in the multiobjective problem. Consider two nodes connected with an arc as in Fig. 2.

The solutions which are associated with these two nodes are:

$$Z_{12} = c_1 + c_2$$
 and

It is clear that the relationship of these two solutions depends only on the sign the coefficient c, has.

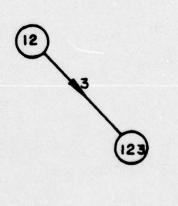


Figure 2

If $c_3>0$ then the solution Z_{12} dominates the solution Z_{123} and if $c_3<0$, Z_{12} is dominated by the solution Z_{123} . Consider now two objective functions Z and W and suppose that the coefficient $c_{13}>0$ (for the function Z), while $C_{23}<0$ (for the function Z). No dominating relation can be established between the two pairs of solutions Z_{12} , Z_{12} and Z_{123} , Z_{123} since Z_{123} and Z_{123} and Z_{123} .

Of course if c was non-negative for all i the solution (12) would dominate the solution (123) and if the first one was feasible , there would be no reason to test the second one.

The above example easily can be extended to the general case and the following theorem can be established.

THEOREM 1. In a multiobjective problem, the solution which is associated with some node "a" in a level k, dominates the solution of some successor of node "a", node "aj", in the next level k+1, if and only if the coefficients c ij of the objective functions which are associated with the index j, are nonnegative for all i.

The proof of this theorem follows directly from the above discussion and will be cmitted.

The notation is illustrated in Fig. 3.

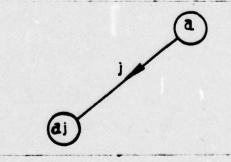


Figure 3

and the strength

The next theorem, a direct result of Theorem 1, provides useful insight into the problem.

THEOREM 2. In a multiobjective problem, a solution associated with a node "a"="ij...", such that, there exist c <0 for some m, c <0 for some l, ni lj and generally there exists some negative coefficient associated with each one cf the digits which form the node "a", cannot be dominated from any other solution in the graph, so it must be tested.

PROOF. The proof will be illustrated with the example in Fig. 4.

The node "123" can be formulated either from node "12" and the digit 3 associated with the arc which connects the two nodes, or from nodes "13" or "23" and the digits 2 or 1 correspondingly. If the digits 1,2 and 3 are all associated with some negative coefficient, it follows directly from

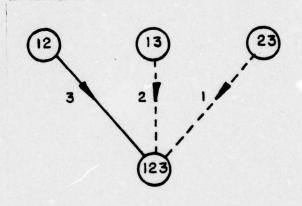


Figure 4

theorem 1 that no predecessor solution dominates the solution "123", so it must be tested.

The problem will now be reformulated and a tree-search algorithm to solve it will be developed.

C. FORMULATION OF THE PROBLEM

The following two transformations are required in order to formulate the problem in a desired form.

- a) If for a given j c ≤0 for all i, then substitute
 ij
 x =1-y to make the coefficients nonnegative.
- b) If c <0 for some i and c >0 for some l and c ≥0 ij lj ik for all i, fcrmulate the problem so that j<k. In other words shift the negative coefficients to the left, by renaming the variables or reassigning subscripts.</p>

The following notation will be followed in the remainder of this paper when dealing with a problem with n variables, p objective functions and m constraints.

 $SN=\{1,2,...,f\}$ is the set of the first f digits (f<n) for which there exist at least one negative coefficient in some (but not in all) objective functions associated with them.

 $SP=\{f+1,f+2,...,n\}$ is the set of digits which are associated with no negative coefficients.

From Theorem 2, it is necessary to test all nodes which have digits only from the set SN, so when representing the set of the sclutions by a graph, as was done in the one objective case, it is reasonable to consider these nodes as a separate graph. This graph "A" will contain all nodes which are combinations of the first f digits, so it will f have 2 nodes.

A graph "B" with nodes which are combinations of the digits from the set SP can also be constructed.

THEOREM 3. The set of all possible solutions of an integer zero-one multiobjective function problem can be represented as the cartesian product of the nodes of two graphs A and B. Graph A is constructed of all combinations of the digits from the set SN and no dominating relation exists between its nodes. Graph B is constructed from all combinations of the digits from the set SP, and its nodes have the same predessessor-successor relationship as the nodes of the graph which represents the solutions of the problem (1).

Proof. It is clear that graph A has 2^f nodes and that graph B has 2^n nodes.

Their cartesian product is 2ⁿ nodes as is expected for a problem of n variables. No repetition of a node can appears in these products, since nodes from two sets have no element in common; so the 2ⁿ nodes represent the set of all possible solutions for the n-variables problem. The fact that no domination relation exists between the nodes of the graph A is a direct result of Theorem 2. On the other hand the structure of the graph B is analogous of the structure of the graphs for the one-objective function problems, because no negative coefficients are associated with the nodes of this graph.

The cartesian product set, which is the set of all nodes, can be partitioned into 2 subsets, each the product

of a node of graph A with all of the nodes of graph B. The succession graph over the resulting set contains only arcs associated with all nonnegative coefficients; normal dominance tactics may be employed.

This consideration of the problem by two graphs is very interesting and very helpful because it links the multiobjective problem with the one objective case.

This is so because, as is apparent from Theorem 2, in any type of algorithm, all nodes of graph A must be tested; on the other hand as long as graph B has the structure of the graph for the one-objective function problem, all the research which has been done in this area can be used for the multiobjective problem, keeping in mind that here the coefficients can not be aranged in increasing order.

These two graphs provide an indication of the size of the problem. Of course in the worst case, it might be necessary to test all the 2ⁿ solutions, but even in the best case it is necessary to test all the solutions which are associated with the nodes of the graph A, e.g. 2^f solutions. Normally f is a small number, because the objective functions have small inclination between each other.

III. SMALL SCALE PROBLEMS. AN EXAMPLE.

Consider the following example multicriterion problem, expressed in the canonical form set out in the preceding section:

min
$$21 = x_1 + 2x_2 + 2x_3 + 3x_4 + 4x_5$$

min $22 = -2x_1 - x_2 + 2x_3 + x_4 + 3x_5$
subject to: (3)
A: $x_1 + x_2 + x_3 + x_4 + x_5 \ge 1$
B: $-x_1 + 3x_2 + 2x_3 + 2x_4 - 3x_5 \ge 0$
C: $x_1 - x_2 + 2x_3 - x_4 + x_5 \ge 0$
and $x_1 + x_2 + x_3 + x_4 + x_5 \ge 0$

A. TOTAL ENUMERATION

First, all possible 2 = 32 solutions will be explicitly enumerated and some statistics will be calculated in order to provide an indication of the redundancy chtained with the tree-search algorithm in the required work. For this purpose Table 1 has been constructed, where first each constraint is checked for feasibility; if one constraint is not satisfied there is no reason to proceed further. If the solution is feasible, then its value is calculated and stored in the proper column. In the last column are kept the current efficient solutions.

node	solution	fea	sibili	Lty	value	efficnt sltns	
		a	b	C			
012345111122233341111123455554 01234511122223341111112223345552222334455555455554	000001000100101100101101111111111111111	ת אין	ת איש	מאלאלאלאלא איי איי איי איי איי איי איי אי	(2, 2) (3, -3) (4, -1) (4, 1) (6, 2) (5, -1) (6, 1) (7, 2) (8, 4) (9, 6) (101, 5) (112, 3)	(2, 2) (3,-3)	
to	tals	. 32	31	23		24	

TABLE 1

From Table 1 the following statistics are obtained:

- i. Number of examined solutions: 32
- ii. Constraints examined for feasibility: 32+31+23=86
- iii. Calculated values of the obj. functions: 2*18=36
 - iv. Compared solutions for efficiency: 24*2=48

Note that in this problem the solution (12) was found very early and since this solution dominates all other solutions except the solution (3), the number of comparisons for efficiency was reduced to 24. Of course this is not the typical case. Figure 5 illustrates the bounding relationships between the solutions, and the reader easily can verify that generally more comparisons are required in order to obtain the set of efficient solutions { (3) and (12) } for the example problem.

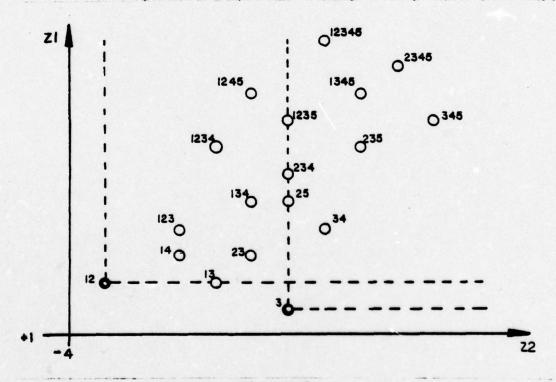


Figure 5

E. THE TREE-SEARCH PROCEDURE

Consider now a tree-search type algorithm in order to reduce the required amount of work. Following the notation introduced for theorem 3:

 $SN=\{1,2\}$ and $SP=\{3,4,5\}$.

As theorem 3 states, the set of all possible solutions will be the cartesian product of the two graphs A and B of Fig. 6.

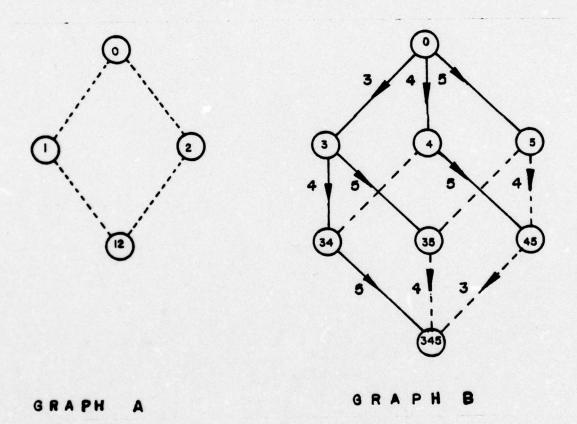


Figure 6

As has been seen, all solutions of graph A must be tested in combination with the nodes of graph B.

Now a logical sequence to visit the nodes of the graphs is as follows:

First a node from the graph A is selected as the rcot of the graph which is constructed from the combination of this node with the nodes of graph B. Then this new graph is searched, testing these nodes which are not bounded. For example consider the composite graph formed from graph B and node (2) of graph A. First solution (2) is tested. If necessary, its immediate successors (23), (24) and (25) are tested; some of their successors may also require testing, and so on.

To reduce testing it is necessary to determine which nodes are dominated. The following rules apply:

RULE 1. If a node is feasible there is no need to test its successors in higher levels.

For example if node (24) is feasible it is unnecessary to test nodes (234), (235) and (2345).

RULE 2. If a solution is dominated by one of the current efficient solutions (i.e. solutions which up to this point have been found to be feasible and not dominated) then there is no need to test its feasibility or to examine its successors.

Let us apply now the preceding to solve the example problem. The procedure is illustrated in Table 2.

node	feas	sibi:	Lity	value	current efficnt	excluded nodes		
	a b c			slns				
0	n							
3	Y	y	y	(2, 2)	(2, 2)	34,35,345		
4	y	У	n	(3, 1)				
5				(4, 3)		45		
1	A	n		(1,-2)				
13	y	У	y	(3, 0)	(3, 0) *	134,135,1345		
14	y	У	У	(4,-1)	(4,-1) *	145		
15				(5, 1)				
2	y	У	n	(2,-1)				
23				(4, 1)		234,235,2345		
24				(5, 0)		245		
25				(6, 2)				
12	У	У	У	(3,-3)	(3,-3)	123,124,125,1234 1235,12345,1245		
total	8	7	6					

TABLE 2

First to be examined are the combinations of the node (0) of graph A with all nodes of graph B, testing for feasibility until finding the first feasible solution. Node (0) is not feasible so node (3) is tested (see also Fig. 6). This node is feasible and according the Rule 1, it is unnecessary to test its successor nodes (34), (35) and (345). From this point on there is a current efficient solution, so when node (4) is examined first its value {3, 1} is calculated and it is determined if it is bounded from the current efficient solution. Because it is not, its feasibility is examined. Since node (4) is not feasible and is not dominated, neither Rule 1 nor Rule 2 can be applied

to exclude its successors from testing. The next node (5), which has a value {4, 3}, is dominated by one of the current efficient values {2, 2}, so it is not necessary to test its feasibility and its successor node 45 can be ignored.

Proceeding in this manner Table 2 is completed, yielding the following statistics:

- i. Number of examined solutions: 13
- ii. Constraints examined for feasibility: 8+7+6=21
- iii. Calculated values of the obj. functions: 2*12=24
 - iv. Compared solutions for efficiency: 19*2=38

Compared with the results from Table 1, here only 40% as many solutions were examined, only 24% of the constraints for feasibility were calculated, and 67% as many values of the objective functions were calculated. Note that here, the solution (12) which dominates most of the other solutions, was the last one which has been tested, while before it was tested very early; nevertheless the number of comparisons made was significantly reduced.

Here are some more rules which improve the efficiency of the algorithm in small scale problems.

RULE 3. Consider a graph B containing a feasible node (a) which dominates another node (b). If some solution (ma) is feasible, there is no need to test the node (mb), since it is dominated from the node (ma).

In the example node 3 dominates node 5. Since node 13 was found to be feasible it was possible to avoid testing node 15.

Single objective implicit enumeration (ref [7]) commonly

takes advantage of the following two observations; since they deal strictly with the constraints, they are directly applicable to multicriterion implicit enumeration as well.

Observasion 1. Consider a node (a) with x =1 if j is in (a) and x =0 otherwise. All the successors of node (a) must have x =1 for j in (a); these variables are fixed for the successors of node (a); all others are said to be free variables as they may take either cf the values 0 or 1. If (a) is not feasible, it is possible that there are not enough free variables left to satisfy a given constraint.

For example assume that a given constraint is $-x_1-x_2+x_3+x_4\ge 1$ and node (12) is under consideration. In this case even with $x_3=x_4=1$ the constraint is still not satisfied. When this happens, there is no need to test the successors of node (12).

Observasion 2. When a subset of variables is fixed, then a given constraint may force some other variables to be fixed also.

For example let a constraint be $2x_1 - x_2 - x_3 \le 0$ in an n-variable problem. Consider node 1. In order to satisfy the given constraint, all the successors of node 1 must have $x_2 = 1$ and $x_3 = 1$. Thus there is no need to test nodes (12) and (13), and among their successors, there is need to test only node (123) and its successors.

IV. LARGE SCALE PROBLEMS

As long as the problem does not have too many variables, one can easily constract the graphs A and B and keep track of the solutions which must be tested or not. But for 0-1 integer problems the number of possible solutions grows exponentially (2^n) with the number of variables (n). Thus for n=5 there are 2^n =32 candidate solutions, but for n=10 there are more than one thousand and for n=30 more than one billion.

A. AN ACCITIVE ALGORITHM

From the above discussion it follows that for large problems, it is necessary to use a procedure to generate those nodes (or solutions), and only those, which must be tested.

The structure of the graphs A and B, and the nature of the problem, suggests an algorithm of additive and/or recursive type. In the example illustrated in the previous section, the procedure followed was to test a node and if one of the bounding rules held, to exclude from testing a set of successor nodes.

Here the procedure is slightly changed. A list of nodes to be checked is maintained, and after testing a given node, if none of the bounding rules apply, the successors of this node in the next level only, are added to the list. The nodes at highest levels are not added since, if it is required for them to be tested, they will be generated when

by the wiley of

their immediate predecessors are investigated. A convenient way to keep track of the nodes which must be generated, in order to be protected from duplications, is as follows:

Consider two nodes with the property that the designation of the second is the designation of the first plus an additional index larger than the larger index of the first. The second node is said to be a direct lexicographic successor of the first. The "successors" of a node include lexicographic successors, their direct successors, and so on. All solutions are partially ordered by this relationship. For the graphs which are considered as the product of a node of graph A with all nodes of graph B, each node dominates its lexicographic successors. of the direct lexicographic succession in figures 1 and 6 are shown as solid lines; they constitute a tree rooted at 0 and spanning all the nodes of the graph.

The above technique permits the calculation of the values of the objective functions and the values of the constraints by the following recursive equations:

Where: Zc: denotes the vector of objective functions

i: denotes the level of the graph

C(j): denotes the vector of coefficients of the objective functions which are associated with x.

Zc: denotes the vector of constraint values.

A(j): denotes the vector of the coefficients of the constraints which are associated with \mathbf{x} .

As an illustration, in the example problem from the previous section, the above values for the node 23 in level 2 are as follows:

Now to calculate these values in the next level 3 for the nodes generated from node 23, nodes 234 and 235, it is only necessary to add the proper coefficients in the values of the previous level. For node 234, C(j)=C(4)=(3, 1) and A(j)=A(4)=(1, 2, -1). Thus:

$$Zc(3) = Zc(2) + C(4) = \{(4+3), (1+1)\} = (7, 2)$$
 and $Zc(3) = Zc(2) + A(4) = \{(2+1), (5+2), (1-1)\} = (3, 7, 0)$

Analogously for the node 235:

$$Zo(3) = (8, 4)$$
 and $Zc(3) = (3, 2, 2)$

The following notation is introduced to help in the formulation of a step by step algorithm which employs these techniques.

SNT1, SNT2 = The sets of nodes to be tested in graphs A
 and B respectively.

SES = The set of currently efficient solutions.

SOH = Solution on hand.

Now the algorithm can be formulated as follows:

STEP 0: (initializations).

SES=empty; SNT2=empty;

SNT1={ d| d is a node from graph A }.

STEP 1:

If SNT1=empty then stop

SNT1=SNT1-ak where ak is some node in SNT1

whose the last digit is k.

SNT2={ ak }

STEP 2: (pick the SOH)

If SNT2=empty go to Step 1

SNT2=SNT2-ak where ak is some ncde in SNT2

STEP 3: (check for dominance)

i. SOH= ak

ii. Calculate Zo for SOH using (4).

iii. If SES is empty go to step 4.

iv. If SOH is bounded by some sclution in SES then SNT2=SNT2-alk for all l=k-1,k-2,...,f+1 and gc to Step 2.

STEP 4: (check for feasibility).

i. Calculate Zc for SOH using (4).

ii. If SOH is not feasible go to Step 5.

iii. Put Zo for SOH in SES.

iv. Eliminate from SES all these solutions which are bounded from SOH.

v. SNT2=SNT2-alk for all l=k-1,k-2,...,f+1.
Go to step 2.

STEP 5: (Generate next level successors)

SNT2=SNT2U {akj|j=k+1,k+2,...,n if k>f

j=f+1,f+2,...,n if k≤f}.

Go to step 2.

State Winds

snt2	3,4,5	5,45 empty	13,14,15		23,24,25	[
Ses	(2,2)			(2,2), (3,0) (2,2), (4,-1)		(2, 2), (3, -3)
b s f		 e		 > >	·	~
ZC	(0, 0, 0)	2,-	(1,-1, 1)	(2, 1, 3)	m ((2, 2, 0)
0 0		n ×		e e >	6 7 7 7	=
0 2	(0,0)	(3, 1)	(1,-2)	(6, -1)	(6, 2)	(3,-3)
snt2	0 smpty 4,5	1 65	l ampty	14,15 15 empty 2	empty 24,25 25 empty 12	empty
snt1	1,2,12		1 2,12	2	empty	stop
soh	o m	3 10	- ;	E # 5	23 24 24 25	2
begn step	- 7 7	7 7	- 7	70 0-	0000-	~ -

The second second

Table 3 summarizes the application of this algorithm to the previously used example.

STEP 0: SES=empty; SNT1={0,1,2,12}; SNT2=empty

STEP 1: SNT1={1,2,12} (Node ak is node 0). SNT2={ 0 }.

STEP 2: SNT2 is not empty so SNT2=SNT2-0=empty.

STEP 3: i. SOH=0

ii. Zo(0) = (0,0)

iii. SES is empty so go to Step 4.

STEP 4: i. Zc(0) = (0,0,0)

ii. Since B=(1,0,0), SOH is not feasible.

Go to Step 5.

STEP 5: We have k=0 and f=2 so $SNT2=\{3,4,5\}$ Go to Step 2.

STEP 2: $SNT2=SNT2-(3)=\{4,5\}$ (We examine solution 3).

STEP 3: i. SOH= (3)

ii. Zo(1) = Zo(0) + C(3) = (0+2, 0+2) = (2,2).

iii. SES=empty so go to Step 4.

STEP 4: i. Zc(1) = Zc(0) + A(3) = (1,2,2).

ii. Since B=(1,0,0), SOH is feasible.

iii. $SES=\{(2,2)\}.$

Go to Step 2.

STEP 2: SNT2={ 5 } (we examine now node (4)).

STEP 3: i. SOH= (4).

ii. Zo(1) = (0+3,0+1) = (3,1).

iv. SOH is not bounded.

STEP 4: i. Zc(1) = (0+1, 0+2, 0-1) = (1, 2, -1).

ii. SOH is not feasible. Go to Step 5.

STEP 5: We have k=4 so $SNT2=SNT2U\{45\}=\{5,45\}$. Go to Step 2.

STEP 2: SNT2={ 45 } (We test now ncde 5).

STEP 3: i. SOH= (5)

ii. Zo(1) = (4, 3)iv. SOH is bounded from the solution (2,2).

SNT2=SNT2-(45)=empty (ak=5 and 1=k-1=4).

Go to Step 2.

STEP 2: SNT2=empty so go to Step 1.

From this point the reader should not have any difficulty following the way Table 3 has been filled in.

Note that in the construction of Table 3, in order to calculate the values of Zo and Zc for a node with more than two digits, two or more values (if they appear in the table) are added. So for example, to calculate the value of Zc for the node (23), the corresponding values of Zc for the nodes (2) and (3) are added. The calculation of the value of Zo is analogous.

So, for the nodes for which the required information already appears in the table, the equations (4) can be replaced by:

where aj is the node with last digit j and the rest of the digits in the string a. Note that j can also be considered as a string of digits and that the digits which form the node can be partitioned in more than two substrings, for which the values of the corresponding nodes must be added to calculate the values for the examined node.

P. IMPLEMETATION USING THE COMPUTER

In this chapter the structure a computer program must have to solve the multiobjective problem will be examined.

In the previous sections the values of Zc and Zc were calculated recursively using the equations (4), and the next level successors of each node were lexicographically generated by successively concatenating to the node all digits which are greater than its last digit. Working with the computer it is important to keep in storage only that information which is required to proceed with the following steps. There are two approaches to search a graph and to generate its nodes.

One approach is to search the graph level by level. First all the information which corresponds to the level 1 is stored. From this information for the n nodes, the information for the next level $\binom{n}{2}$ nodes is produced and

kept in memory in order to generate the information for the next level, and so on. Another approach is to search the graph depth first, keeping in storage one only node from each level. When the generated node has its last digit equal to n, the procedure backtracks in the previous level and the graph is searched again all the way down until a node with a last digit of n is generated. As an example, in a problem with 5 variables the nodes 0,1,12,123,1234 and 12345 are first generated, and then the procedure backtracks and replaces in level 4 node 1234 with the next successor of node 123, node 1235. Since again the last digit is equal to n, the procedure goes back two levels and from node 12 generates nodes 124 and 1245; from here two levels back

again and from node 12 generates now node 125, and so on. In the first approach, the number of nodes which must be kept in storage changes from level to level and its maximum value is $\binom{n}{1}$ where l=n/2 for n even or $l=n/2\pm1/2$ for n odd. In the second approach the number is constant and it is equal to n+1. Table 4 gives an indication of these numbers.

n	5	6	7	10	15	20
(n)	10	20	35	252	6435	184756
	6					

TABLE 4

In the FORTRAN program of the appendix, the second approach has been used.

STORE WALLE

APPENDIX A

The computer program for the implementation of the algorithm has been written in FORTAN, the most popular language for the Operation Researchers. The algorithm for this program is nearly that described in the previous section; the sets are implemented as arrays which are searched and updated when required. A block structured language such as PASCAL or PLI, permitting the use of sets and array comparisons, would allow a clearer, more faithful representation.

The program consists of the main program and the subroutine CHILD. In the main program, the solution 0 is first tested for feasibility. If it is feasible, the next node is generated from graph A (node 1), and the value of the solution 0 is added to the set of efficient sclutions If the solution 0 is not feasible, its first successor node is generated from graph B (i.e. the node which represents the digit f+1). This is the initial step. From now on the recursive equations (4) can be used since the values of Zo and Zc in level 0 are both zero. After the successor to node 0 has been generated either in graph A or in graph B, the subroutine CHILD is called to test this node and to return to the main program the order to generate (IGNRT=1) or to not generate (IGNRT=0) its successors (children). Depending on the value of the parameter IGNRT, the main program searches, always depth first, the graphs and generates the next solution to be tested by the subroutine CHILD. From the main program two parameters are

passing to the subroutine. The parameter j corresponds to the level of the node from which the examined node has been generated, and the parameter m indicates the last digit in the examined node.

The subroutine CHILD works as follows. First the values the objective function for the examined node are calculated. If the SES is empty, the feasibility of the node is examined and if it is feasible IGNRT=0 is returned. Otherwise the order to generate the child is given to the The subroutine continues to test for program. feasibility, until finding the first feasible solution. This solution is added to SES and from that point the program tests first if the examined node is dominated by some node in SES and then, if it is not, it tests its feasibility. If a solution is not dominated and it is feasible, then it is added in to SES and the solutions which are dominated by the new solution are removed from SES. variable NSES keeps track of the number of solutions which are in SES. The values of the variables for the solutions in SES are stored in the array X1, so that the program is able to print both the values of the objective functions and the solutions to which they correspond.

Using this program the example problem has been solved and the nodes which have been tested are printed, so that the reader to can compare these results with the ones obtained from Table 3. The only difference here is that node 45 has been tested; in Table 3 the fact that node 5 was bounded, eliminated the need to test the node 45. The reason is that as long as the graphs are tested depth first, after node 4 has been tested node 45 is generated first and then node 5.

It must be noticed that this program can be used to solve one objective function problems also; the variable NO,

which corresponds to the number of the objective functions, is given the value 1.

The results from 17 problems run on the IBM 360 computer of N.P.S. have been summarized in Table 5.

1		1		1		1		1	number	1	number of	1	cpu	ŧ
1	n	1	na	1	no	1	nc	1	of slns	1	efficient	1	time	1
1		1		1		1		ł	tested	1	solutions	1	sec.	1
1_		_1_		_1_		_1_		_1.		_1.		1.		_1
1	5	1	2	1	2	1	3	1	14	1	2	1	0.24	1
1	6	1	0	1	3	1	4	1	21	1	1	1	0.33	1
1		1	2	1	2	1	4	1	64	1	0	1	0.66	1
1		1	2	1	3	1	4	1	64	ı	0	1	0.68	1
1		1	2	1	4	1	4	1	58	1	5	1	0.80	1
1		1	3	1	3	1	6	1	58	1	3	1	0.64	1
1	9	1	0	1	3	1	4	1	163	1	2	t	1.91	1
1		1	3	1	3	1	5	1	346	1	4	1	3.67	1
1		1	3	1	2	ſ	4	1	377	1	5	1	3.39	1
1		1	3	1	3	1	4	1	97	1	4	1	1.14	1
1		1	3	1	4	1	4	1	2 5 3	i	13	1	3.68	- 1
1	10	1	0	1	1	1	7	1	388	1	1	1	3.43	1
1		1	0	1	3	1	4	1	175	1	3	1	2.42	1
1		1	3	1	2	1	4	1	224	1	2	1	2.23	1
1		1	3	1	3	1	4	1	232	1	8	1	2.74	1
1		1	4	1	3	1	4	1	331	1	2	1	4.19	1
1	25	1	5	1	2	1	3	1	747	1	7	1	2.53	i
1		1	5	1	2	1	3	1	12049	1	7	1	14.72	1
1_		_1_		_1_		_1_		_1.		_1.		.1.		_1

TABLE 5

A STATE OF THE STA

THIS PROGRAM SO VES A MULTI OR ONE-CEJECTIVE FUNCTION INTEGER ZERO-CNE MINIMIZATION PROBLEM, FORMALIZED IN THE CANONICAL FORM.

CREATOR: AGGELOS C. SIMOPOULOS MAJOR HELLENIC ARMY DECEMBER 1977

THE FOLLOWING NOTATION HAS BEEN USEC:

N = NUMBER OF VARIABLES IN THE PROBLEM
A = NUMBER OF VARIABLES ASSOCIATED WITH AT LEAST ONE
NEGATIVE COEFFICIENT
B = N-NA
C = NUMBER OF OBJECTIVE FUNCTIONS
C = NUMBER OF CONSTRAINTS NA =

NB = NO = NC =

20(N+1,NO) = ARRAY CCNTAINING THE VALUES CF THE CBJ. FUNCTIONS IN EACH LEVEL

ZC(N+1,NC) = ARRAY CCNTAINING THE VALUES CF THE CROSSTRAINTS IN EACH LEVEL

SES(**,NO) = SET OF AT MOST ** EFFICIENT SLNS

IFLSES = FLAG; INDICATES IF SES IS EMPTY OR NOT IFLAG1 = FLAG; INCICATES IF SLN C IS FEASIBLE CR NOT NSES = CURRENT NUMBER CF SLNS IN SES

X1(**,N) = CONTAINS THE VALUES OF THE VARIABLES FCR

** SOLUTIONS IN SES

NX1 = CURRENT NUMBER OF SLNS IN X1

LCA(NA+1) = LAST DIGIT OF A NODE IN GRAPH EACH LEVEL OF THE GRAPH LCB(NB+1) = AS ABOVE FOR GRAPH "B" 11 4 11 FOR

CURRENT LAST DIGIT IN A CURRENT LEVEL CF GRAPH CURRENT LEVEL CF GRAPH =

C(NC,N) = MATRIX OF COEFFICIENTS A(NC,N) = MATRIX OF COEFFICIENTS E(NC) = THE E VECTOR FOR THE FOR THE O. FUNCTIONS
FOR THE CONSTRAINTS
CONSTRAINT EQUATIONS

THE FOLLGWING PROGRAM HAS BEEN ARRANGED TO SOLVE PROBLEMS UP TO 25 VARIABLES, 5 CBJECTIVE FUNCTIONS AND 10 CONSTRAINT EQUATIONS. CEPENDING ON THE NUMBER OF THE VARIABLES N. FORMATS 1201 AND 1202 MUST BE CHANGED EACH TIME TO NF5.0 AND (N+1)F5.0 CORRESPONDINGLY TO GIVE SUITABLE CUTPUT.

IF AN ERROR OCCURS AND THE USER HAS CHECKED TO BE SURE THAT THE INPUT DATA IS ACCORDING TO THE FORMAT OF THE PROGRAM, THEN THE FIRST CIMENSION OF THE ARRAYS X1 AND SES MUST BE INCREASED. FOR BIGGER PROBLEMS THE DECLARATIONS AND FORMATS MUST BE CHANGED ACCORDING TO THE NOTATION GIVEN ABOVE.

STEEL MINE

```
CCC
                    BLCCK DATA
INTEGER
DIMENSION
CCMMON /S/
                                                       X(25)
ZO(26,5),ZC(26,10)
X,IFLSES,IFLAGI,NX1,NSES,MA,LA,LE,ZG,ZC
X,IFLSES,IFLAGI,NX1,NSES,MA,LA,LE,ZC,ZC
/29*0,3*1,390*0./
                 1
                     END
COCOCOCO
                                                              X(25),X1(150,25),LDA(6),LDB(26)
ZO(26,5),ZC(26,10),C(5,25),A(10,25),B(10),
SES(150,5)
X,IFLSES,IFLAG1,NX1,NSES,MA,LA,LB,ZO,ZC
SES,N,NO,NC,C,A,B,X1
                     INTEGER
DIMENSION
                     CCMMON /S/
CCMMON /S1/
CCCCC
                                                                     N, NA, NO, NC
((C(I, J), J=1, N), I=1, NO)
((A(I, J), J=1, N), I=1, NC)
(B(I), I=1, NC)
                    READ
READ
READ
READ
                                      (5,100C)
(5,1001)
(5,1001)
(5,1002)
                    WRITE(6,110C)
WRITE(6,1201)
WRITE(6,1101)
WRITE(6,1202)
WRITE(6,1105)
                                                                      ((C(I,J),J=1,N),I=1,NO)
                                                                      ((A(I,J),J=1,N),B(I),I=1,NC)
CCCC120001
10001
10001
10001
11200
                                            (/6X,25(I3))
(415)
(10F8.3)
(10F8.3)
(5F5.0)
(6F5.0)
(//12X, 'COEFFICIENT MATRIX FOR THE',
(//12X, 'COEFFICIENT MATRIX FOR THE',
'CONSTRAINTS'//)
(///12X, 'THE SLNS FOR THE ABOVE PROBLEM ARE')
(//5X, 'VALUE:',5F8.2)
(/12X, 'SOLUTION:',25I4)
(///12X, 'THE FCLLOWING SLNS HAVE BEEN TESTED')
(///12X, 'NO FEASIBLE SOLUTION WAS FOUND')
                    FCRMAT
FCRMAT
FCRMAT
FCRMAT
FCRMAT
FCRMAT
FCRMAT
 1101
                     FCRMAT
                     FCRMAT
FCRMAT
FCRMAT
FCRMAT
FCRMAT
MISCELLANEOUS
                               (NA.EQ.O) NA=1
(NA.EQ.N) NA=N-1
CCC
```

```
TEST SOLUTION O FIRST
            WRITE (6,12CO) (X(I),I=1,N)
DO 2 I=1,NC
IF (E(I).GT.O.) GO TO 20
CONTINUE
200000
          OTHERWISE SOLUTION O IS FEASIBLE
            DO 3 I=1,N

X1(1,I)=0

CONTINUE

N×1=1

DO 4 I=1,NO

SES(1,I)=0.

CONTINUE

NSES=1

IFLSES=1
3
4
               IFLAG1=1
LA=1
NOU O
          GENERATE NODES FROM GRAPH A
            DC 30 JA=LA,NA

MB=NA+1

IF (IFLAG1.NE.1) GD TO 120

LB=JA+1

DO 22 II=MB,N

X(II)=0

CONTINUE

X(MA)=1

WRITE(6,1200) (X(I),I=1,N)

CALL CHILD (JA,MA,IGNRT)

IF (IGNRT.EG.0) GD TO 25
        22
CCCC120
130
138
C
139
                  IF (MB.EG.N) GO TO 135
X(MB)=0
LB=JB
MB=MB+1
GO TO 12C
```

```
00005
             RETURN TO GRAPH A
                 IF (IFLAG1.NE.1) GO TO 19
LCA(JA)=MA
IF (MA.EQ. NA) GO TO 35
MA=MA+1
CCNTINUE
LA=JA-1
IF (LA.EC.O) GO TO 80
LLA=LDA(LA)
CO 38 KA=LLA,NA
X(KA)=0
CONTINUE
MA=LLA+1
30
38
                  MA=LLA+1
GC TO 20
80000
             PRINT OUT THE SOLUTIONS
               85
95
000000000000
          THE SUBROUTINE CHILD TESTS A SOLUTION FOR DOMINATION AND FEASIBILITY AND RETURNS TO THE MAIN PROGRAM THE CROER TO GENERATE OR NOT GENERATE THE SUCCESSORS OF THE CORRESPONDING NODE
               SLBROUTINE CHILD (J,M,IGNRT)

INTEGER X(25),X1(150,25)

DIMENSIGN ZO(26,5),ZC(26,10),C(5,25),A(10,25),B(10),

SES(150,5)

CCMMON /S/ X,IFLSES,IFLAG1,NX1,NSES,MA,LA,LB,ZO,ZC

CCMMON /S1/ SES,N,NO,NC,C,A,B,X1
C
                         DO 202 I=1,NO
ZO(J+1,I)=ZO(J,I)+C(I,M)
CONTINUE
202
C
               IF (IFLSES.EQ.O) GO TO 210
        TEST IF THE SLN IS DCMINATED
                         DO 205 K=1, NSES

ICNTR=0

DO 204 I=1, NC

IF (ZO(J+1,I).LT.SES(K,I)) GO TC 205

IF (ZO(J+1,I).EQ.SES(K,I)) ICNTR=ICNTR+1

CONTINUE

IF (ICNTR.EQ.NO) GO TO 205

CONTINUE

CONTINUE

CONTINUE
204
205
```

```
CCC210
       CHECK FOR FEASIBILITY
                        DO 212 I=1,NC

ZC(J+1,I)=ZC(J,I)+A(I,M)

CONTINUE

DO 215 I=1,NC

IF (ZC(J+1,I).LT.B(I)) GC TO 230

CONTINUE
212
215
               IF (IFLSES.EQ. 0) GO TO 315
COCOC
       UPDATE SES AND X1
              N21=NSES+1
DO 302 I=1,N
X1(NX1,I)=X(I)
CONTINUE
302
CC
CC
CC
CC
CC
        ELIMINATE COMINATED SLNS FROM SES AND X1
              K=1
                       DO 310 I=1.NSES

ICNTR=0

DO 305 II=1.NC

IF (SES(I:II):LT.ZG(J+1:II)) GO TC 306

IF (SES(I:II):EQ.ZO(J+1:II)) ICNTR=ICNTR+1
                                  CCNTINUE
IF (ICNTR.EQ.NO) GO TO 306
GO TO 310
305
CC 206
       CTHERWISE KEEP THE SLNS IN SES AND IN X1
                       DO 308 IK=1,NC
SES(K,IK)=SES(I,IK)
CONTINUE
DO 309 IN=1,N
CONTINUE
K=K+1
CONTINUE
CONTINUE
DO 312 IK=1,NC
SES(K,IK)=ZO(J+1,IK)
CONTINUE
DO 313 IN=1,N
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
SES(K,IK)=X1(NX1,IN)
CONTINUE
SES(K,IN)=X1(NX1,IN)
308
309
310
312
313
              CONTINUE

NSES=K
GC TO 220

DO 317 IK=1,NG

SES(1, IK) = ZO(J+1, IK)

CONTINUE

NSES=1

IFLSES=1

OO 318 I=1, N

X1(1, I) = X(I)

NX1=1
315
317
318
Ç
220
               IGNRT=0
RETURN
230
               IGNRT=1
               ENC
```

The state of

COEFFITIENT MATRIX FOR THE CEJECTIVE FUNCTIONS

-1: -2: 2: 3: 4: -2: -1: 2: 1: 3:

COEFFICIENT MATRIX FOR THE CONSTRAINTS

THE FOLLOWING SLNS HAVE BEEN TESTED

0000

C 0 1 0 0

0 0 1 0

0 0 1 1

0 0 0 1

.

1 0 1 0 0

1 0 6 1 0

.

. 1 . 0 1

THE SLAS FOR THE ABOVE PROBLEM ARE

VALUE: 2.00 2.00

SOLUTION: C O 1 O C

VALUE: 3.00 -3.00

SOLUTION: 1 1 0 0 C

LIST OF REFERENCES

- 1. Zangwill, Willard, Non Linear Programming: A Unified Approach, Prentice Hall 1967.
- Beale, E.M.L., On Quadratic Programming, p. 227 to
 Research Logistics Quartely, Vol 6, 1959.
- Baukol J. and P. Wolfe, A Warehouse Location Problem,
 p. 252 to 263, Operations Research, Vol 6, 1958.
- 4. Dorfman r., P.A. Samuelson and R. Solow, Linear Programming And Economical Analysis, Mc Graw Hill 1958.
- 5. Gass, Saul I., Linear Programming: Methods And Applications, Mc Graw Hill, Third Edition, 1969.
- 6. Orchard Hays, William, Advanced Linear Programming Computing Techniques, Mc Graw Hill, 1968.
- Harvey M. Wagner, <u>Principles Of Operations Research</u>,
 p. 385 to 388, Prentice Hall, Inc., 1969.
- 8. Claude Mc Millan, Jr, <u>Mathematic al Frogramming: An Introduction To The Design And Application Of Ortimal Desision Machines</u>, p. 425 to 444, Appendix c, John Wiley And Sons, Inc., 1970.
- 9. Stanley Zionts, An Interactive Method For Evaluating Descrete Alternatives Involving Multiple Criteria, Working Paper No. 271, University Of New York At Eufalo School Of Management, July 1976.
- 10. Gabriel R. Bitran, Linear Multiple Objective Programs With Zero-One Values, p. 121 to 139, Mathematical Programming, Vol 13, 1977.

and the wine

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center	2
	Cameron Station	
	Alexandria, Virginia 22314	
2.	Library, Code 0142	2
	Naval Postgraduate School	
	Monterey, California 93940	
3.	Department Chairman, Code 52	2
	Department of Computer Science	
	Naval Postgraduate School	
	Monterey, California 93940	
4.	Department Chairman, Code 55	2
	Department of Operations Research	
	Naval Postgraduate School	
	Monterey, California 93940	
5.	Lcdr Stephen T. Holl, Code 52	3
	Departement of Computer Science	
	Naval Postgraduate School	
	Monterey, California 93940	
6.	Greek Army Headquarters	2
	Department Of Research And Design	
	Holargos, Athenes, GREECE	
7.	Major Aggelos C. Simopoulos	3
	Kourtiou 29,	
	1+hens COR CPPPCP	